CSC 347 - Concepts of Programming Languages

Formal Semantics

Instructor: Stefan Mitsch

Learning Objectives

❓ How to unambiguously define the semantics of a programming language?

- Identify the difference between syntactic and semantic expressions in operational semantics
- Identify judgments and reduction rules
- Apply reduction rules to execute a program

A Small Programming Language

Language Constructs

- Integer expressions
- Statements
  - Assignment
  - Statement lists
  - Conditional
  - Loop

Example Program: Factorial

```
n := −5;
if n>=0
  then i := n
  else i := 0−n
fi;
f := 1;
while i do
  f := f * i;
  i := i−1
od
```

Operational Semantics

- Operational semantics defines meaning of programs relative to an *abstract machine*
- **Reduction machine**: operates on a program and reduces it to its semantic "value"
  - Uses a store ξ (e.g., a map from variables to values)
  - State of the machine is a program or value and the store $\langle \mathbf{\texttt{prg}}, \xi \rangle$ or $\langle v, \xi \rangle$
  - Judgments: $\langle \mathbf{\texttt{prg}}, \xi \rangle \Downarrow \langle v, \xi' \rangle$

    > Executing program $\mathbf{\texttt{prg}}$ in state ξ *yields* ($\Downarrow$) value v and new state ξ'

  - Reduction rules: $\frac{\text{premise}_1, \ldots, \text{premise}_n}{\text{conclusion}}$

    > Conclusion follows from all premisses satisfied

Language of Integer Expressions

Syntax

```
Expr ::= Number
       | Expr '+' Expr
       | Expr '-' Expr
       | Expr '*' Expr
       | '(' Expr ')'
```

Example

```
2+3*4
(2+3) * 4
```

Semantics

- Semantic domain: integer arithmetic
- Value v is the meaning of an expression
  - For example, `(2 + 3) * 4` means 20
- Numbers evaluate to their value

$$\frac{}{\langle \mathtt{n},\xi\rangle \Downarrow \langle n,\xi\rangle}\ (\text{Num})$$

- Operations map to integer operations

$$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \mathtt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{e}_1\mathtt{+}\mathtt{e}_2,\xi_0\rangle \Downarrow \langle v_1+v_2,\xi_2\rangle}\ (\text{Add}),\ \ldots,\ \frac{\langle \mathtt{e},\xi\rangle \Downarrow \langle v,\xi_1\rangle}{\langle \mathtt{(e)},\xi\rangle \Downarrow \langle v,\xi_1\rangle}\ (\text{Par})$$

Reduction Example: Arithmetic Expression

Rules

- $\dfrac{}{\langle \texttt{n},\xi\rangle \Downarrow \langle n,\xi\rangle}\ (\mathrm{Num})$

- $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{e}_1\texttt{+}\texttt{e}_2,\xi_0\rangle \Downarrow \langle v_1+v_2,\xi_2\rangle}\ (\mathrm{Add})$

- $\dfrac{\langle \texttt{e},\xi\rangle \Downarrow \langle v,\xi_1\rangle}{\langle \texttt{(e)},\xi\rangle \Downarrow \langle v,\xi_1\rangle}\ (\mathrm{Par})$

- Reduce 2 + 3 to its semantic value
- Deduction tree

$$\dfrac{\dfrac{}{\langle 2,\xi\rangle \Downarrow \langle 2,\xi\rangle}(\mathrm{Num}) \qquad \dfrac{}{\langle 3,\xi\rangle \Downarrow \langle 3,\xi\rangle}(\mathrm{Num})}{\langle 2+3,\xi\rangle \Downarrow \langle 5,\xi\rangle}(\mathrm{Add})$$

- Define the rule for multiplication
- Reduce `(2 + 3)` $*$ 4 to its semantic value
- $$\dfrac{\dfrac{\dfrac{\dfrac{}{\langle 2,\xi\rangle \Downarrow \langle 2,\xi\rangle}(\mathrm{Num}) \qquad \dfrac{}{\langle 3,\xi\rangle \Downarrow \langle 3,\xi\rangle}(\mathrm{Num})}{\langle 2+3,\xi\rangle \Downarrow \langle 5,\xi\rangle}(\mathrm{Add})}{\langle \texttt{(}2+3\texttt{)},\xi\rangle \Downarrow \langle 5,\xi\rangle}(\mathrm{Par}) \qquad \dfrac{}{\langle 4,\xi\rangle \Downarrow \langle 4,\xi\rangle}(\mathrm{Num})}{\langle \texttt{(}2+3\texttt{)} * 4,\xi\rangle \Downarrow \langle 20,\xi\rangle}(\mathrm{Mul})$$

Assignments and Sequential Composition

Syntax

```
Expr   ::= Number
         | Expr '+' Expr
         | Expr '-' Expr
         | Expr '*' Expr
         | Ident
         | PrgSeq
         | '(' Expr ')'
Ident  ::= 'a' | 'b' | ... | 'z'
PrgSeq ::= Prg | Prg ';' PrgSeq
Prg    ::= Ident ':=' Expr
```

Example

```
a := 2+3;
b := (a:=a+1)*4;
a := b-5
```

Semantics

? How do we store/look up values of variables?

- Look up variable in store

$$\frac{}{\langle \mathtt{x}, \xi \rangle \Downarrow \langle \xi(\mathtt{x}), \xi \rangle}\ (\mathrm{Var})$$

- Evaluate expression in current state, then update state

$$\frac{\langle \mathtt{e}, \xi_0 \rangle \Downarrow \langle v, \xi_1 \rangle}{\langle \mathtt{x} := \mathtt{e}, \xi_0 \rangle \Downarrow \langle v, \xi_1\{\mathtt{x} \mapsto v\} \rangle}\ (:=)$$

- Evaluate subexpressions and chain results

$$\frac{\langle \mathtt{e}_1, \xi_0 \rangle \Downarrow \langle v_1, \xi_1 \rangle \qquad \langle \mathtt{e}_2, \xi_1 \rangle \Downarrow \langle v_2, \xi_2 \rangle}{\langle \mathtt{e}_1 ; \mathtt{e}_2, \xi_0 \rangle \Downarrow \langle v_2, \xi_2 \rangle}\ (;)$$

Conditionals and Loops

Syntax

```
Expr   ::= Number
         | Expr '+' Expr
         | Expr '-' Expr
         | Expr '*' Expr
         | Ident
         | Expr '>=' Expr
         | PrgSeq
         | '(' Expr ')'
Ident  ::= 'a' | 'b' | ... | 'z'
PrgSeq ::= Prg | Prg ';' PrgSeq
Prg    ::= Ident ':=' Expr
         | 'if' Expr
            'then' Expr
            'else' Expr 'fi'
         | 'while' Expr 'do'
            Expr
           'od'
```

Semantics

- Conditional
  - True: $$\frac{\langle \mathtt{e}_1,\xi_0\rangle\Downarrow\langle v_1,\xi_1\rangle \qquad v_1\neq 0 \qquad \langle \mathtt{e}_2,\xi_1\rangle\Downarrow\langle v_2,\xi_2\rangle}{\langle \texttt{if } \mathtt{e}_1 \texttt{ then } \mathtt{e}_2 \texttt{ else } \mathtt{e}_3 \texttt{ fi},\xi_0\rangle\Downarrow\langle v_2,\xi_2\rangle}\ (\text{iftrue})$$
  - False: $$\frac{\langle \mathtt{e}_1,\xi_0\rangle\Downarrow\langle v_1,\xi_1\rangle \qquad v_1=0 \qquad \langle \mathtt{e}_3,\xi_1\rangle\Downarrow\langle v_2,\xi_2\rangle}{\langle \texttt{if } \mathtt{e}_1 \texttt{ then } \mathtt{e}_2 \texttt{ else } \mathtt{e}_3 \texttt{ fi},\xi_0\rangle\Downarrow\langle v_2,\xi_2\rangle}\ (\text{iffalse})$$
- Loop
  - End: $$\frac{\langle \mathtt{e}_1,\xi_0\rangle\Downarrow\langle v_1,\xi_1\rangle \qquad v_1=0}{\langle \texttt{while } \mathtt{e}_1 \texttt{ do } \mathtt{e}_2 \texttt{ od},\xi_0\rangle\Downarrow\langle 0,\xi_1\rangle}\ (\text{whileend})$$
  - Recurse: $$\frac{\langle \mathtt{e}_1,\xi_0\rangle\Downarrow\langle v_1,\xi_1\rangle \qquad v_1\neq 0 \qquad \langle \mathtt{e}_2;\texttt{while } \mathtt{e}_1 \texttt{ do } \mathtt{e}_2 \texttt{ od},\xi_1\rangle\Downarrow\langle v_2,\xi_2\rangle}{\langle \texttt{while } \mathtt{e}_1 \texttt{ do } \mathtt{e}_2 \texttt{ od},\xi_0\rangle\Downarrow\langle v_2,\xi_2\rangle}\ (\text{whilerec})$$

Program Reduction Example: Absolute Value

```
i := -2;
if i>=0
  then i := i
  else i := 0-i
fi
```

- Integer arithmetic
  - $$\frac{}{\langle \mathtt{n},\xi\rangle \Downarrow \langle n,\xi\rangle}\ (\text{Num})$$
  - $$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \mathtt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{e}_1\mathtt{+}\mathtt{e}_2,\xi_0\rangle \Downarrow \langle v_1-v_2,\xi_2\rangle}\ (\text{Sub})$$
- Assignment
  - $$\frac{\langle \mathtt{e},\xi_0\rangle \Downarrow \langle v,\xi_1\rangle}{\langle \mathtt{x} \mathtt{:=} \mathtt{e},\xi_0\rangle \Downarrow \langle v,\xi_1\{\mathtt{x}\mapsto v\}\rangle}\ (\mathtt{:=})$$
- Sequential composition
  - $$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \mathtt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{e}_1\,\mathtt{;}\,\mathtt{e}_2,\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}\ (;)$$
- Comparison
  - $$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad \langle \mathtt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{e}_1\mathtt{>=}\mathtt{e}_2,\xi_0\rangle \Downarrow \langle \max(0,v_1-v_2+1),\xi_2\rangle}\ (\geq)$$
- Conditional
  - $$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1\neq 0 \qquad \langle \mathtt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3\ \mathtt{fi},\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}\ (\text{iftrue})$$
  - $$\frac{\langle \mathtt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1=0 \qquad \langle \mathtt{e}_3,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \mathtt{if}\ \mathtt{e}_1\ \mathtt{then}\ \mathtt{e}_2\ \mathtt{else}\ \mathtt{e}_3\ \mathtt{fi},\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}\ (\text{iffalse})$$

Program Reduction Example: Absolute Value

$$
\dfrac{
\dfrac{\dfrac{}{\langle -2, \{\} \rangle \Downarrow \langle -2, \{\} \rangle}\,(\text{Num})}{\langle \texttt{i := } -2 \rangle \Downarrow \langle -2, \{\texttt{i} \mapsto -2\} \rangle}\,(:=)
\qquad
\dfrac{\text{Lemma } \texttt{i >= 0} \qquad 0 = 0 \qquad \text{Lemma } \texttt{i := 0} - \texttt{i}}{\langle \texttt{if i >= 0 then i := i else i := 0} - \texttt{i fi} \rangle \Downarrow \langle 2, \{\texttt{i} \mapsto 2\} \rangle}\,(\text{iffalse})
}{\langle \texttt{i := } -2\texttt{; if i >= 0 then i := i else i := 0} - \texttt{i fi}, \{\} \rangle \Downarrow \langle 2, \{\texttt{i} \mapsto 2\} \rangle}\,(;)
$$

- Lemma $\texttt{i >= 0}$, where $\xi = \{\texttt{i} \mapsto -2\}$
- Lemma $\texttt{i := 0} - \texttt{i}$, $\xi = \{\texttt{i} \mapsto -2\}$

$$
\dfrac{\dfrac{}{\langle \texttt{i}, \xi \rangle \Downarrow \langle -2, \xi \rangle}\,(\text{Var}) \qquad \dfrac{}{\langle \texttt{0}, \xi \rangle \Downarrow \langle 0, \xi \rangle}\,(\text{Num})}{\langle \texttt{i >= 0}, \xi \rangle \Downarrow \langle 0, \xi \rangle}\,(\geq)
$$

$$
\dfrac{\dfrac{\dfrac{}{\langle \texttt{0}, \xi \rangle \Downarrow \langle 0, \xi \rangle}\,(\text{Num}) \qquad \dfrac{}{\langle \texttt{i}, \xi \rangle \Downarrow \langle -2, \xi \rangle}\,(\text{Var})}{\langle \texttt{0} - \texttt{i}, \xi \rangle \Downarrow \langle 2, \xi \rangle}\,(\text{Sub})}{\langle \texttt{i := 0} - \texttt{i}, \xi \rangle \Downarrow \langle 2, \xi\{\texttt{i} \mapsto 2\} \rangle}\,(:=)
$$

Program Reduction Example: Factorial

```
n := -5;
if n>=0
  then i := n
  else i := 0-n
fi;
f := 1;
while i do
  f := f*i;
  i := i-1
od
```

- Integer arithmetic
  - $\dfrac{}{\langle \texttt{n},\xi\rangle \Downarrow \langle n,\xi\rangle}$ (Num)
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{e}_1 - \texttt{e}_2,\xi_0\rangle \Downarrow \langle v_1 - v_2,\xi_2\rangle}$ (Sub)
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{e}_1 * \texttt{e}_2,\xi_0\rangle \Downarrow \langle v_1 \cdot v_2,\xi_2\rangle}$ (Mul)
- Assignment
  - $\dfrac{\langle \texttt{e},\xi_0\rangle \Downarrow \langle v,\xi_1\rangle}{\langle \texttt{x := e},\xi_0\rangle \Downarrow \langle v,\xi_1\{\texttt{x}\mapsto v\}\rangle}$ (:=)
- Sequential composition
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \quad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{e}_1\,\texttt{;}\,\texttt{e}_2,\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}$ (;)
- Conditional
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{e}_1\texttt{>=}\texttt{e}_2,\xi_0\rangle \Downarrow \langle \max(0,v_1-v_2+1),\xi_2\rangle}$ ($\geq$)
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1 \neq 0 \qquad \langle \texttt{e}_2,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{if e}_1\ \texttt{then e}_2\ \texttt{else e}_3\ \texttt{fi},\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}$ (iftrue)
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1 = 0 \qquad \langle \texttt{e}_3,\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle}{\langle \texttt{if e}_1\ \texttt{then e}_2\ \texttt{else e}_3\ \texttt{fi},\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}$ (iffalse)
- Loop
  - $\dfrac{\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1 = 0}{\langle \texttt{while e}_1\ \texttt{do e}_2\ \texttt{od},\xi_0\rangle \Downarrow \langle 0,\xi_1\rangle}$ (whileend)
  - $\dfrac{\begin{array}{c}\langle \texttt{e}_1,\xi_0\rangle \Downarrow \langle v_1,\xi_1\rangle \qquad v_1 \neq 0 \\ \langle \texttt{e}_2\texttt{;while e}_1\ \texttt{do e}_2\ \texttt{od},\xi_1\rangle \Downarrow \langle v_2,\xi_2\rangle\end{array}}{\langle \texttt{while e}_1\ \texttt{do e}_2\ \texttt{od},\xi_0\rangle \Downarrow \langle v_2,\xi_2\rangle}$ (whilerec)

Program Reduction Example: Factorial

$$
\dfrac{
\dfrac{\dfrac{}{\langle -5, \{\}\rangle \Downarrow \langle -5, \{\}\rangle}\,(\text{Num})}{\langle \texttt{n}{:=}\ {-5}, \{\}\rangle \Downarrow \langle -5, \{\texttt{n} \mapsto -5\}\rangle}\,(:=)
\qquad
\dfrac{\text{Lemma } \texttt{if...fi} \qquad \dfrac{\text{Lemma } \texttt{f} := \texttt{1} \qquad \text{Lemma } \texttt{while...od}(5)}{\langle \texttt{f} := \texttt{1; while...od}, \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 5\}\rangle \Downarrow \langle 0, \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}\,(;)}{\langle \texttt{if...fi; f := 1; while...od}, \{\texttt{n} \mapsto -5\}\rangle \Downarrow \langle 0, \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}\,(;)
}{\langle \texttt{n}{:=}\ {-5}\texttt{; if...od}, \{\}\rangle \Downarrow \langle 0, \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}\,(;)
$$

- Lemma $\texttt{if...fi}$, where $\xi = \{\texttt{n} \mapsto -5\}$

$$
\dfrac{
\dfrac{\dfrac{}{\langle \texttt{n}, \xi\rangle \Downarrow \langle -5, \xi\rangle}\,(\text{Var}) \qquad \dfrac{}{\langle \texttt{0}, \xi\rangle \Downarrow \langle 0, \xi\rangle}\,(\text{Num})}{\langle \texttt{n >= 0}, \xi\rangle \Downarrow \langle 0, \xi\rangle}\,(\geq)
\qquad 0{=}0 \qquad
\dfrac{\dfrac{\dfrac{}{\langle \texttt{0}, \xi\rangle \Downarrow \langle 0, \xi\rangle}\,(\text{Num}) \qquad \dfrac{}{\langle \texttt{n}, \xi\rangle \Downarrow \langle -5, \xi\rangle}\,(\text{Var})}{\langle \texttt{0}{-}\texttt{n}, \xi\rangle \Downarrow \langle 5, \xi\{\texttt{i} \mapsto 5\}\rangle}\,(\text{Sub})}{\langle \texttt{i} := \texttt{0}{-}\texttt{n}, \xi\rangle \Downarrow \langle 5, \xi\{\texttt{i} \mapsto 5\}\rangle}\,(:=)
}{\langle \texttt{if n >= 0 then i := n else i := 0}{-}\texttt{n fi}, \xi\rangle \Downarrow \langle 5, \xi\{\texttt{i} \mapsto 5\}\rangle}\,(\text{iffalse})
$$

- Lemma $\texttt{f} := \texttt{1}$, where $\xi = \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 5\}$

$$
\dfrac{\dfrac{}{\langle \texttt{1}, \xi\rangle \Downarrow \langle 1, \xi\rangle}\,(\text{Num})}{\langle \texttt{f} := \texttt{1}, \xi\rangle \Downarrow \langle 1, \xi\{\texttt{f} \mapsto 1\}\rangle}\,(:=)
$$

Program Reduction Example: Factorial

- Lemma $\texttt{while...od}(5)$, where $\xi = \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 5, \texttt{f} \mapsto 1\}$

$$\dfrac{\dfrac{}{\langle \texttt{i}, \xi\rangle \Downarrow \langle 5, \xi\rangle}(\text{Var}) \qquad 5 \neq 0 \qquad \dfrac{\text{Lemma } \texttt{f := f * i} \qquad \dfrac{\text{Lemma } \texttt{i := i - 1} \qquad \text{Lemma } \texttt{while...od}(4)}{\langle \texttt{i := i - 1; while i do ... od}, \xi\{\texttt{f} \mapsto 5\}\rangle \Downarrow \langle 0, \xi\{\texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}(;)}{\langle \texttt{f := f * i; i := i - 1; while i do ... od}, \xi\rangle \Downarrow \langle 0, \xi\{\texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}(;)}{\langle \texttt{while i do f := f * i; i := i - 1 od}, \xi\rangle \Downarrow \langle 0, \xi\{\texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}\rangle}(\text{whilerec})$$

- Lemma $\texttt{f := f * i}$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{f}, \xi\rangle \Downarrow \langle 1, \xi\rangle}(\text{Var}) \qquad \dfrac{}{\langle \texttt{i}, \xi\rangle \Downarrow \langle 5, \xi\rangle}(\text{Var})}{\langle \texttt{f * i}, \xi\rangle \Downarrow \langle 1 \cdot 5, \xi\rangle}(\text{Mul})}{\langle \texttt{f := f * i}, \xi\rangle \Downarrow \langle 5, \xi\{\texttt{f} \mapsto 5\}\rangle}(:=)$$

- Lemma $\texttt{i := i - 1}$

$$\dfrac{\dfrac{\dfrac{}{\langle \texttt{i}, \xi\rangle \Downarrow \langle \xi(\texttt{i}), \xi\rangle}(\text{Var}) \qquad \dfrac{}{\langle \texttt{1}, \xi\rangle \Downarrow \langle 1, \xi\rangle}(\text{Num})}{\langle \texttt{i - 1}, \xi\rangle \Downarrow \langle 5 - 1, \xi\rangle}(\text{Sub})}{\langle \texttt{i := i - 1}, \xi\rangle \Downarrow \langle 4, \xi\{\texttt{i} \mapsto 4\}\rangle}(:=)$$

- Lemma $\texttt{while...od(0)}$, where $\xi = \{\texttt{n} \mapsto -5, \texttt{i} \mapsto 0, \texttt{f} \mapsto 120\}$

$$\dfrac{\dfrac{}{\langle \texttt{i}, \xi\rangle \Downarrow \langle 0, \xi\rangle}(\text{Var}) \qquad 0 = 0}{\langle \texttt{while i do f := f * i; i := i - 1 od}, \xi\rangle \Downarrow \langle 0, \xi\rangle}(\text{whileend})$$

Summary

- **Operational semantics**: defines language in terms of operations of an abstract machine
  - Alternative semantics definitions: denotational semantics, axiomatic semantics
- **Judgments** and **reduction rules**: describe steps of the abstract machine, expressed as conclusions from premises
- **Deduction tree**: makes conclusions about program from the meaning of the program's components